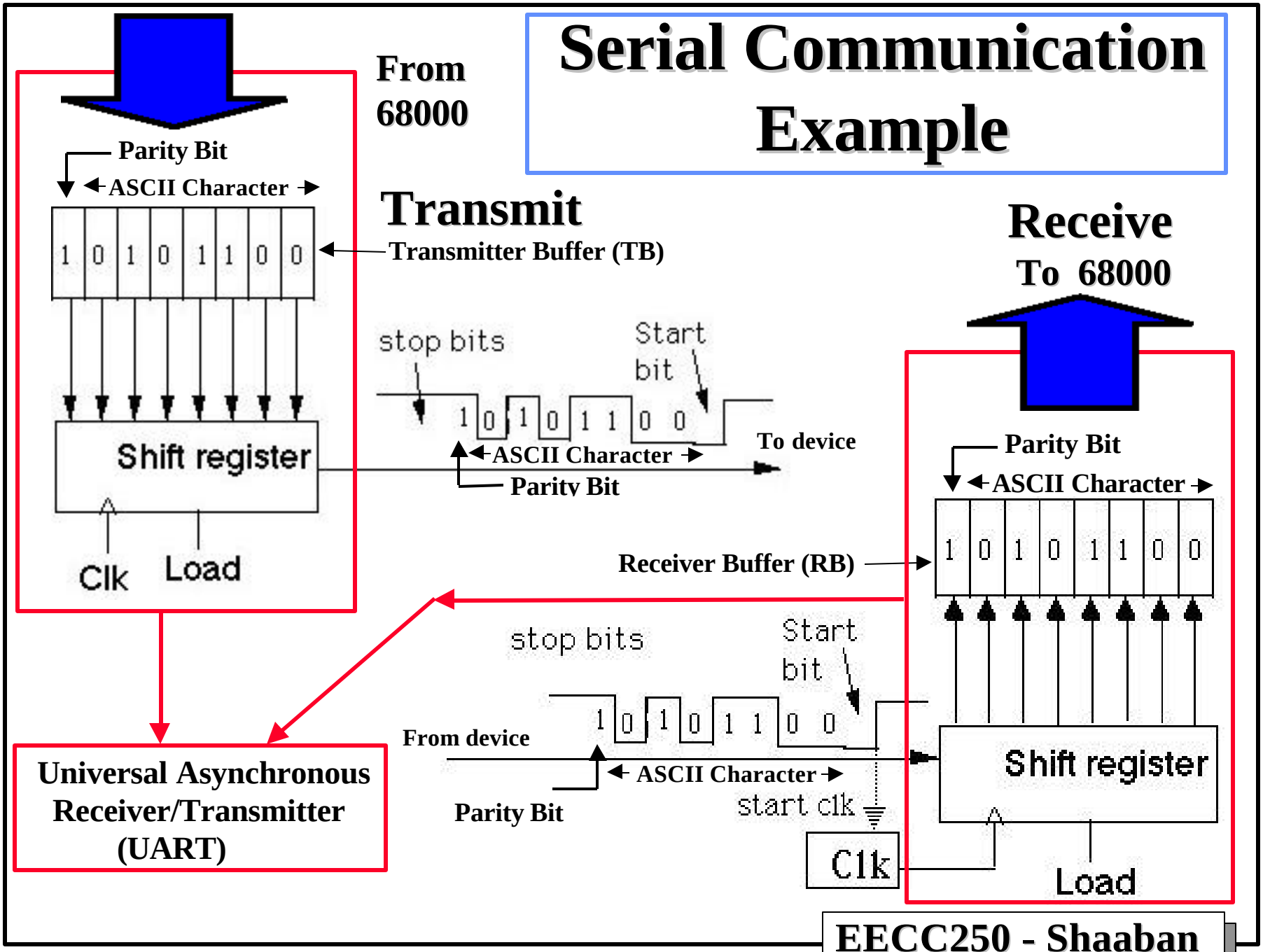
Computer Input and Output (I/O)

- **One of the basic and essential features designed in a computer system is its ability to exchange data with other external devices, and to allow the user to interact with the system:**
  - **Input Devices include:**
    - **Switches, Keyboards, Mice, Scanners, Cameras, etc.**
  - **Output devices include:**
    - **Lamp/LED/LCD displays, Video monitors, Speakers, Printers, etc.**
- **One or more interface circuits usually are used between I/O devices and the CPU to:**
  - **Handle transfer of data between CPU and I/O interface.**
  - **Handle transfer of data between I/O device and interface.**
  - **Enable the CPU to request the status of data sent/received by the interface.**
- **Common I/O interfaces:**
  - **Serial I/O: RS-232C Data exchanged one bit at a time.**
  - **Parallel I/O: Date exchanged one byte at a time.**

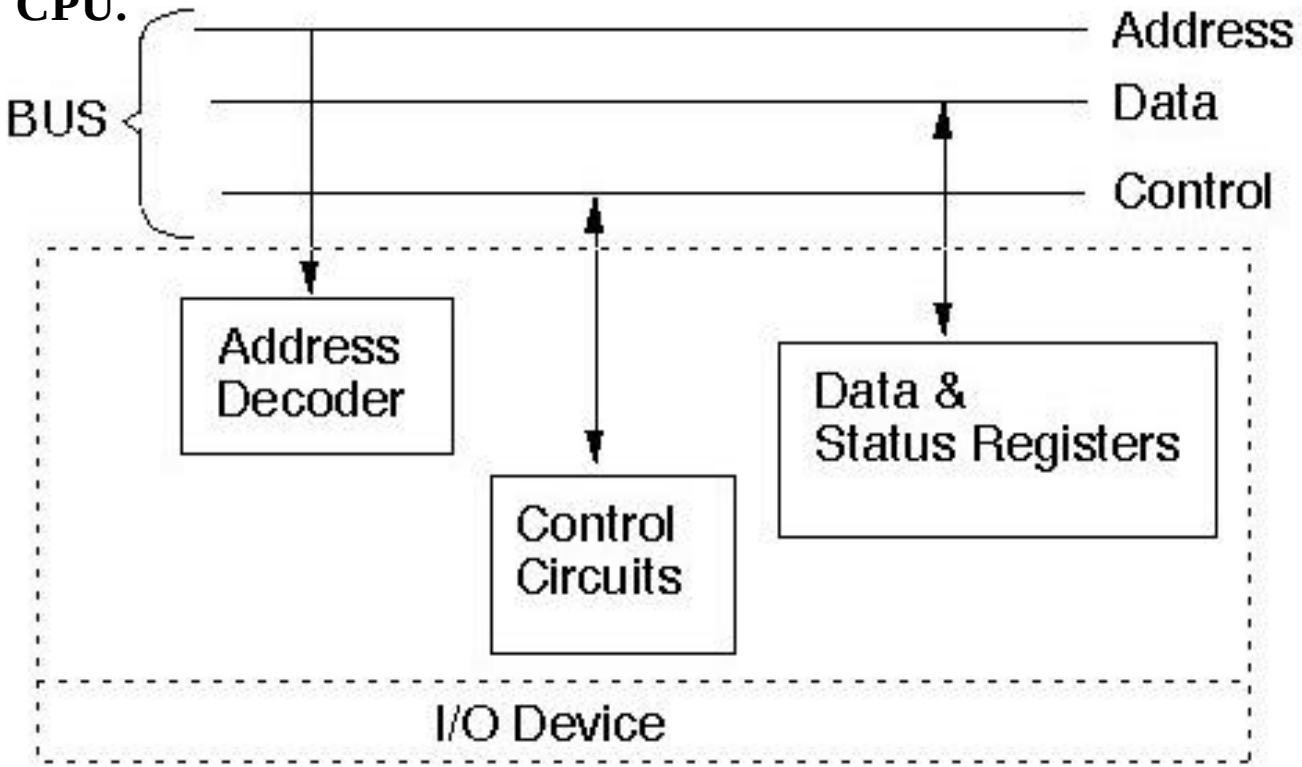
Serial Communication Example



A Typical CPU I/O Connection

For each I/O device or interface:

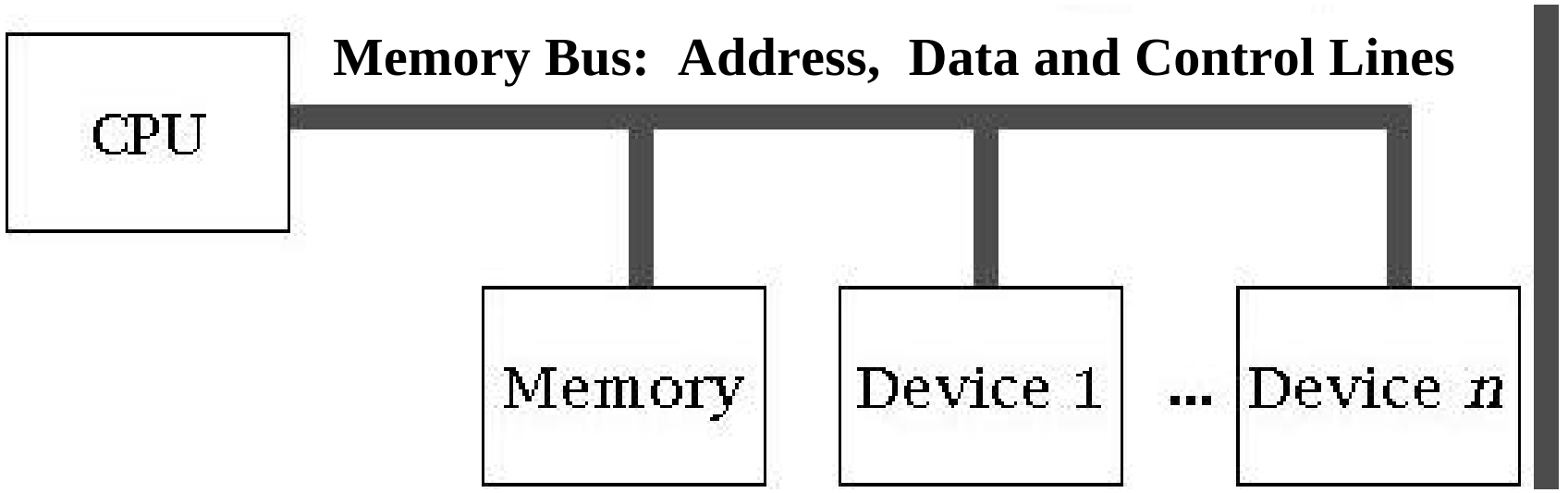
- **A number of registers, reachable by the CPU, are present.**
- **These registers are used for data transfer, I/O device control and configuration and for device status monitoring by the CPU.**
- **Each of the registers is given a unique address.**
- **The address decoder enables the device to recognize its addresses when issued by CPU.**



Memory Mapped I/O

Addresses of data, control and status registers in I/O devices or interfaces are treated by the CPU as if they were conventional memory locations or addresses:

Hence the same instructions that move data to or from memory can be used to transfer data to or from I/O devices.



68000 Memory Mapped I/O

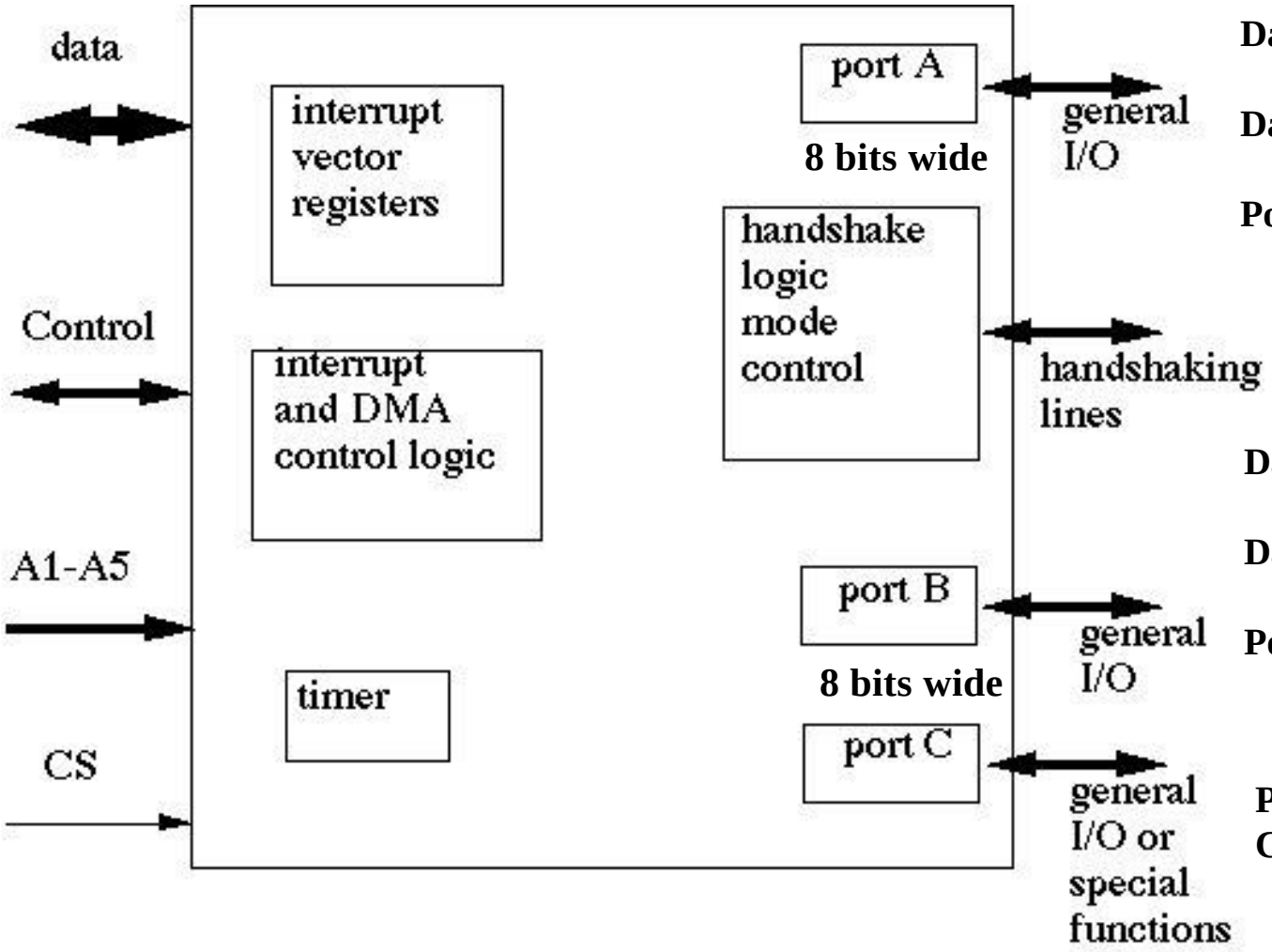
The Motorola 68000 uses memory mapped I/O, where device registers are assigned unique addresses within the memory address space. I/O data and control registers are treated as if they were memory locations.

Example: The SBC08K 68008 board used in the lab includes:
Two parallel ports A, B using the Motorola 68230 Parallel Interface/Timer (PI/T) chip, with a Port General Control Register, PGCR address of $0FF000

- **Parallel data port A (or PA) has the following addresses:**
  - **Data Register of port A, DRA has address: $0FF010**
  - **Data Direction Register of port A, DDRA has address: $0FF004**
  - **Port A Control Register, PACR has address: $0FF00C**
- **Parallel data port B (or PB) has the following addresses:**
  - **Data Register of port B, DRB has address: $0FF012**
  - **Data Direction Register of port B, DDRA has address: $0FF006**
  - **Port B Control Register, PBCR has address: $0FF00E**

The Motorola 68230 Parallel Interface Timer (PI/T)

- **A general purpose Parallel Interface and Timer, PI/T chip that offers several *very complex* modes of operation.**



Data Register,
DRA: $0FF010
Data Direction Register,
DDRA: $0FF004
Port A Control Register,
PACR: $0FF00C

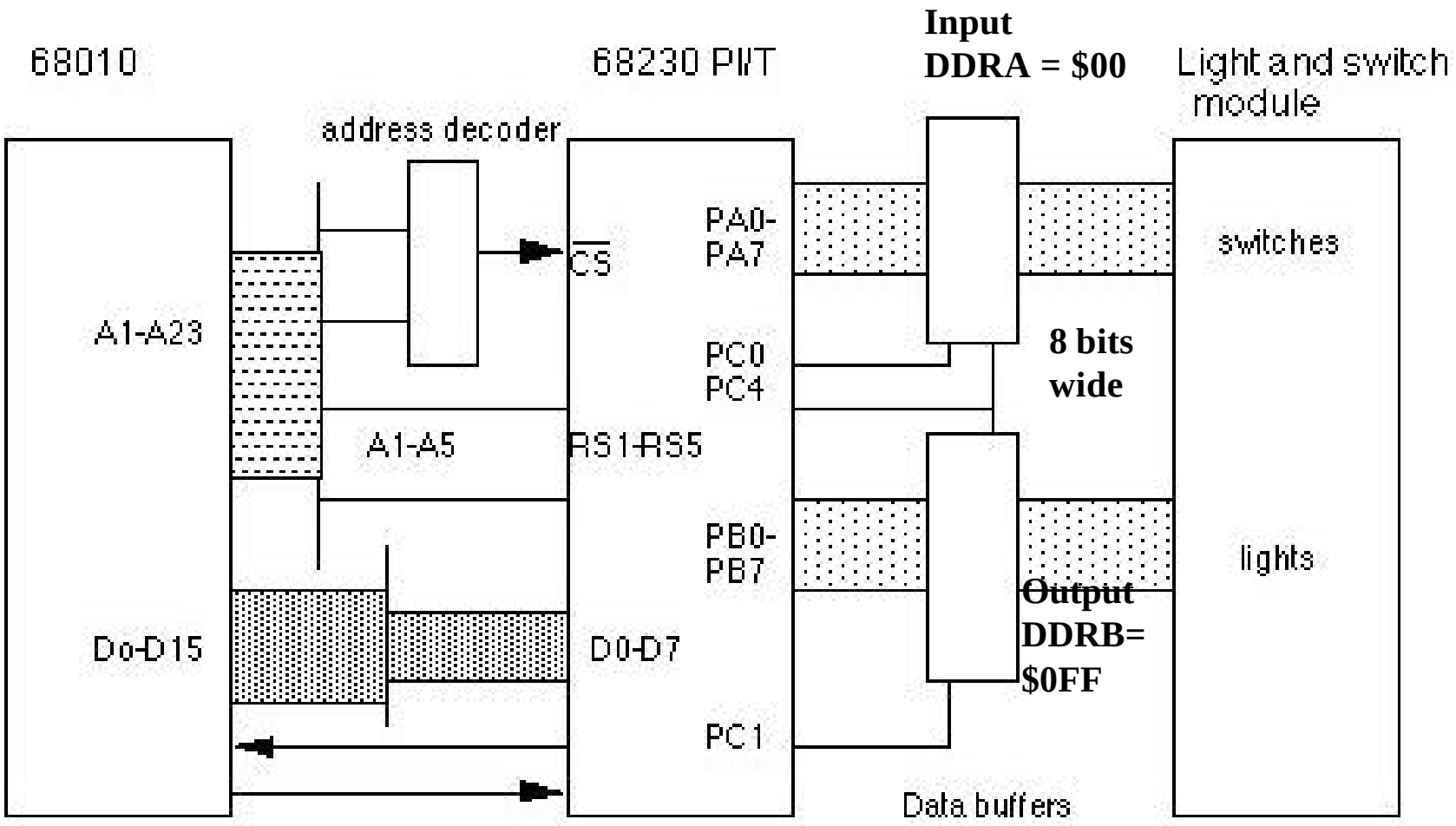
Data Register,
DRB: $0FF012
Data Direction Register:
DDRB: $0FF006
Port B Control Register,
PBCR: $0FF00E

Port General Control Register
PGCR: $0FF000

The Motorola 68230 PI/T

- Contains three 8-bit parallel ports: PA, PB & PC
- PA & PB can be programmed as input or output ports, or as both at the same time (full-duplex operation).
- Can be programmed to interrupt the processor when any port receives new data.
- 68230 also contains a programmable 24 bit counter.
- Handshaking lines can be programmed to provide different communications protocols to the I/O device.
- The 68230 is programmed, and data transfers take place using a total of 23 internal 8-bit registers.

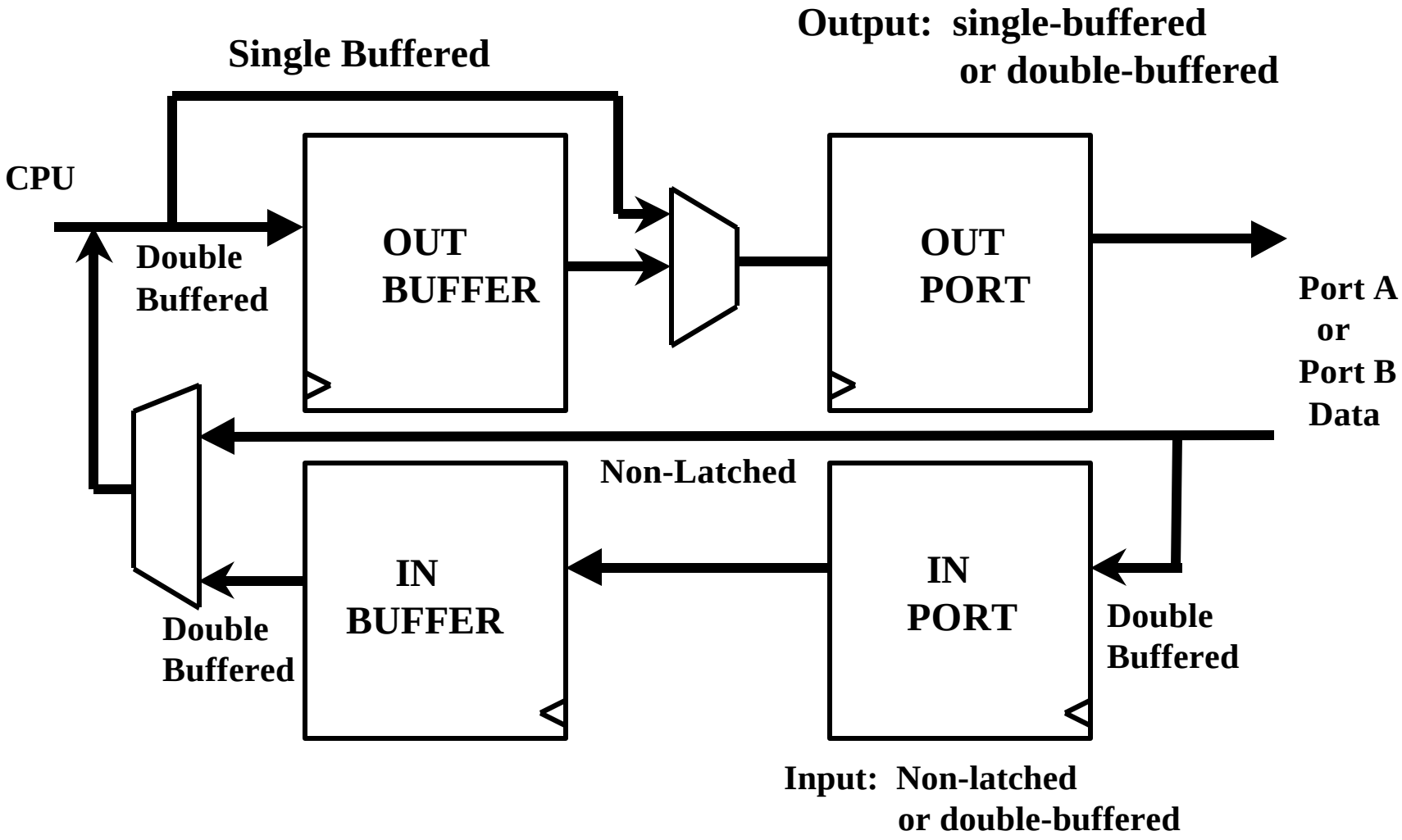
A Typical 68230 Single Board Setup



Programming The 68230

- **Ports A and B are capable of operating in one of four possible modes programmed using the two msb's of PGCR:**
  - **Mode 0 Unidirectional 8 bit transfers (used in lab 3 PGCR = 00).**
  - **Mode 1 Unidirectional 16 bit transfers (PA is MSB).**
  - **Mode 2 Bidirectional 8 bit transfers.**
  - **Mode 3 Bidirectional 16 bit transfers.**
- **Within each of these modes are sub modes programmed using PACR and PBCR:**
  - **00 Double-buffered input, single buffered output.**
  - **01 Double buffered output, no latching of inputs.**
  - **1X Input unlatched, No buffering of output.**
- **Each of the three ports has a Data Direction Register (DDRA, DDRB and DDRC) associated with it:**
  - **Each bit in the DDR controls the direction of I/O on the corresponding bit on the port (1 for output and 0 for input).**
  - **e.g. DDRA = $00 for input $FF for output.**

68230 Parallel I/O Data Latching/Buffering



68230 Register Address Equates

```
PIT     EQU   $0FF000    Base Address of PI/T
PGCR    EQU   PIT        Port General Control Register
PSRR    EQU   PIT+2      Port service request register
PADDR   EQU   PIT+4      Data direction register A
PBDDR   EQU   PIT+6      Data direction register B
PACR    EQU   PIT+$0C    Port A control register
PBCR    EQU   PIT+$0E    Port B control register
PADR    EQU   PIT+$10    Port A data register
PBDR    EQU   PIT+$12    Port B data register
PSR     EQU   PIT+$1A    Port status register
TCR     EQU   PIT+$20    Timer control register
TSR     EQU   PIT+$34    Timer status register
```

I/O Example This program continuously reads Port A (e.g. switches) and outputs the value to port B (LEDs)

```
        ORG     $1000
DRA     EQU     $0FF010     Data Register of Port A
DDRA    EQU     $0FF004     Data Direction Register of Port A
PACR    EQU     $0FF00C     Port A Control Register
DRB     EQU     $0FF012     Data Register of Port B
DDRB    EQU     $0FF006     Data Direction Register of Port B
PBCR    EQU     $0FF00E     Port B Control Register
PGCR    EQU     $0FF000     Address of Port General Control Register
PGCRM   EQU     $00         Equate set mode to 0
DDA     EQU     $00         Equate Port A direction:  input
DDB     EQU     $FF         Equate Port B direction: Output
START   MOVE.B  #PGCRM,PGCR Initialize the mode to  0
        MOVE.B  #$80,PACR   Initialize port A to submode 1x,  non-latched
        MOVE.B  #$80,PBCR   Initialize Port B to mode 1x, single buffered
        MOVE.B  #DDA,DDRA   Initialize Port A as input port
        MOVE.B  #DDB,DDRB   Initialize Port B as output port
LOOP    NOP                 No operation
        MOVE.B  DRA,D0      Read a byte from Port A into D0
        MOVE.B  D0,DRB      Write a byte to Port B  (value read from A)
        NOP                 No operation
        BRA     LOOP        Always  branch.
        END     START
```